The Report Committee for Jiachao Fang
Certifies that this is the approved version of the following report:

Stance Classification in Social Media using Machine Learning Techniques

APPROVED BY
SUPERVISING COMMITTEE:

Matthew Lease, Supervisor

Unmil P. Karadkar

Stance Classification in Social Media using Machine Learning Techniques

by

Jiachao Fang

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Information Studies

The University of Texas at Austin

May 2019

Acknowledgments

First and foremost, I want to thank Dr. Matthew Lease, supervisor of my project, to invite me to claim checker research group, which inspires my research interest in this area. Without his guidance and assistance through the whole process, I could not have completed this project.

I want to thank Dr. Unmil P. Karadkar for being faculty reader for my master's report.

I also want to thank Dr. An T. Nguyen for helping me in all aspects of the project. I received great help from An during the whole process of the capstone, from baseline reproduction, model implementation to report writing. Without him, the report could not be completed so smoothly.

Finally, I would like to thank my family and my friends for their support during the process of my capstone. Without their understanding, I could not have finished this report.

Abstract

Stance Classification in Social Media using Machine Learning Techniques

Jiachao Fang, M.S.INFO.ST

The University of Texas at Austin, 2019

Supervisor: Matthew Lease

Stance classification has been a popular research topic. The target of my research is to build a model to classify post stance in social media, which can later be used to determine the veracity of rumors in social media. My research mainly consists four parts: related work review, baseline reproduction, model exploration and optimization, and results and analysis. All the posts are classified into four categories: support, deny, query and comment. I first tried to reproduce the baseline. Then, I explored and evaluated the performances of different models, and compared model performances. Finally, I summarized the results and gave future research directions.

Table of Contents

Acknowledgments iv

Abstract v

List of Tables ix

List of Figures x

Chapter 1. Introduction 1

1.1 Task description . . . 2
1.2 Report structure . . . 2
1.3 Results . . . 3

Chapter 2. Related Work 5

2.1 Natural language processing . . . 5
2.2 Text classification . . . 6
2.3 Stance classification . . . 9
2.4 Fact checking in social media . . . 10

Chapter 3. Rumour Evaluation Dataset 11

3.1 Dataset exploration . . . 11
3.2 Evaluation metrics . . . 12
3.2.1 Confusion matrix . . . 13
3.2.2 Accuracy score . . . 13
3.2.3 Macro F1 . . . 14
3.2.3.1 Precision . . . 15
3.2.3.2 Recall . . . 15
3.2.3.3 F1 score . . . 15

Chapter 4. Baseline Exploration **17**
4.1 Baseline model . . . 18
4.2 Experiments . . . 18
4.2.1 Experiment I: running with a local machine . . . 19
4.2.2 Experiment II: hyperparameter searching . . . 21
4.2.3 Experiment III: results . . . 23
4.3 Lessons learned . . . 25
4.3.1 Why baseline models? . . . 26
4.3.2 Lessons learned . . . 26
4.3.2.1 Time management . . . 26
4.3.2.2 Always have a plan B . . . 27
4.3.2.3 Theory versus practice . . . 27

Chapter 5. Concepts and models **29**
5.1 Concepts . . . 29
5.1.1 Supervised learning . . . 29
5.1.2 Imbalanced data . . . 30
5.1.3 Overfitting and underfitting . . . 31
5.2 Models . . . 32
5.2.1 SVMs . . . 32
5.2.2 Logistic Regression . . . 33
5.2.3 Decision Tree . . . 35
5.2.4 Random Forest . . . 37

Chapter 6. Experiments and results **38**
6.1 Experiments design . . . 38
6.2 Results . . . 39
6.2.1 Experiment I . . . 39
6.2.1.1 SVMs . . . 39
6.2.1.2 Logistic Regression . . . 39
6.2.1.3 Decision Tree . . . 40
6.2.1.4 Random Forest . . . 40
6.2.1.5 Summary . . . 40

6.2.2 Experiment II 42
6.2.3 Conclusion 43
6.3 Future directions 44

Appendix **45**

Appendix 1. Reflections **46**
1.1 Planning and research design 46
1.2 Baseline exploration 47
1.3 Experiment design and result analysis 47

Bibliography **49**

List of Tables

3.1 Label distribution of training and test dataset 12
3.2 Confusion matrix 13

4.1 Experiment I: environment 20
4.2 Experiment II: environment 21
4.3 Hyperparameter search scope 22
4.4 Comparison of hyperparameter sets 23
4.5 Results 24
4.6 Model performance comparison 24

6.1 Accuracy and macro F1 of SVMs 39
6.2 Accuracy and macro F1 of Logistic Regression 39
6.3 Accuracy and macro F1 of Decision Tree 40
6.4 Accuracy and macro F1 of Random Forest 40
6.5 Results comparison 41
6.6 Results comparison of Random Forest and Branch LSTM . . . 42

1.1 Timeline of the capstone 46

List of Figures

4.1 An example of the thread structure 19

5.1 Sigmoid function 34
5.2 Cost function 35
5.3 An example of the decision tree 36

Chapter 1

Introduction

Fact checking has been a popular research topic. In this era of information explosion, verifying the authenticity of information is vital, yet challenging as technology advances. With the popularity of the social media, the propagation speed of rumours have reached an unprecedented level. Therefore, it is an urgent task for contemporary scientists to conduct researches on rumour veracity, and help to construct a more trustworthy network environment to the society.

The target of my research is to build a model to determine the stance of posts in social media, which can later be used to predict the veracity of rumors in social media. This is a task of International Workshop on Semantic Evaluation 2019, which is Task 7: RumourEval 2019: Determining Rumour Veracity and Support for Rumours. Details can be found in the official website of the competition from https://competitions.codalab.org/competitions/19938.

The tasks are clearly defined by sponsors of this challenge, which include two subtasks: Task A, stance classification of the responses to a rumorous post; Task B, veracity prediction of the statements. My research focuses on

Task A as follows.

1.1 Task description

To verify a rumour in social media, it is crucial to analyze how people response toward the rumour. Therefore, stance classification of the responses to a rumorous post help the analysis of the veracity prediction of the post. In this task, I classified each reply post into the following classes:

1. **Support:** The respondent believes the claim in the original post is true.
2. **Deny:** The respondent believes the claim in the original post is false.
3. **Query:** The respondent is not sure about the truthfulness of the original post, and asks for additional evidence.
4. **Comment:** The comment of the respondent is not related to the veracity of the rumour.

The dataset I used is from the challenge Task 7: RumourEval 2019: Determining Rumour Veracity and Support for Rumours. I will introduce the details of this dataset in Chapter 3.

1.2 Report structure

The report is structured as follows:

In Chapter 1, as introduction, I included task description, report structure and summarizes results and conclusions for the stance classification task.

In Chapter 2, I reviewed the related work, which includes four research areas: natural language processing, text classification, stance classification, and fact checking in social media.

In Chapter 3, I described the rumour evaluation dataset used for stance classification task and selected some suitable metrics for this task according to the characteristics of this dataset.

In Chapter 4, I introduced the baseline model, described my experiments to reproduce the baseline, and concluded my lessons learned.

In Chapter 5, some essential concepts and models used in my stance classification task are presented.

In Chapter 6, I described the experiment design, present, analyzed the experiment results by comparing the model performances and proposed future research directions in social media.

Finally, I attached an appendix of my reflection on the whole capstone process.

1.3 Results

For the stance classification task, I first tried to reproduce baseline, then ran four basic models: SVMs, Logistic Regression, Decision Tree, and Random Forest. Then, I compared their performances and selected a winning solution. Finally, I compared the performances of the baseline model and my winning solutions.

Some outstanding results and conclusions can be summarized as follows:

1. I did several experiments, including changing the system environments, shifting the python version, and re-installing the packages, to reproduce the baseline model, but I could not reproduce the same result as Kochkina, Liakata and Augenstein (2017).

2. For the stance classification tasks, using word2vec and tweet lexicon as input features, of the four basic models I ran, Random Forest, which is the deserved winning solution, outperformed SVMs, Logistic Regression and Decision Tree by 4%.

3. Basic models and neural models have their own advantages and disadvantages, when selecting a suitable model for a specific task, we should consider the properties of the dataset and the target of the task.

Chapter 2

Related Work

In this chapter, I will review the related concepts and work. Since my research focuses on stance classifications in social media, I will explore the related work from a large area, natural language processing, then text classification, and stance classification. Also, I will review related works about fact-checking in social media.

2.1 Natural language processing

Natural language processing generally refers to using machines, usually a computer, to process and understand human languages. Natural language processing is interdisciplinary, which is related to computer science, linguistics, and machine learning.

Natural language processing tasks are usually divided into four categories: syntax, semantics, discourse, and speech [3].

1. Syntax is the structure of the sentences, in which forms, the words are ordered without grammar errors. Typical syntax tasks in natural language processing include: lemmatization, parsing, and stemming.

2. Semantics is to understand the meaning of human languages. Typical semantic tasks in natural language processing include machine translation, natural language generation, and sentiment analysis.

3. Discourse usually refers to the works which focus on the connections of two segments in the sentence. Typical discourse tasks in natural language processing include: automatic summarization, coreference resolution, and discourse analysis.

4. Researches on speech usually involve reciprocal conversion between speech and text. Typical speech tasks in natural language processing include: speech recognition, speech segmentation, and text-to-speech.

Most natural language processing projects is not limited to one task. For example, stance classification may consist of both syntax and semantics tasks, including parsing, stemming, sentiment analysis, and so on.

2.2 Text classification

Text classification, or more general, document classification, is a subarea of computer science, information science, and library science [1].

Traditionally, document classification is subject-oriented; for example, documents can be classified according to its author, title, and type. Generally speaking, there are two types of document classifications: document-based and query-based classification [1].

As people's requirements for exact searching growing, automatic document classification has become the inevitable trend of this area. Automatic document classification is using computer algorithms to classify documents automatically. There are two types of automatic document classifications: supervised document classifications and unsupervised document classifications (document clustering). Supervised classifications classify document with labels, while unsupervised classifications do not know the labels.

My research is essentially a supervised document classification task since I will classify all the posts and responses into four categories (labels): support, deny, comment, query, and comment.

Some popular automatic document classification features are as follows:

1. **N-grams:** Cavnar and Trenkle (1994) define n-grams as "An N-character slice of a longer string" [8]. N-grams approach has the advantage of processing noisy text data, which is generated by OCR systems or other similar sources. Also, N-grams can save the efforts of the stemming, because n-grams features for related words are quite similar (e.g., 'prepare', 'preparation', 'preparing', etc.).

2. **TF-IDF:** TF-IDF refers to term frequency inverse document frequency. Term frequency is the counts a word showed in a document, and inverse document frequency measures the frequency a word showed in all documents. TF-IDF is the multiplication of term frequency and inverse document frequency.

Comparing to other text classification techniques, TF-IDF is easy to use, yet it still has some drawbacks. TF works efficiently with small documents, however, when document size grows, the performance may decrease dramatically [23].

3. **Expectation Maximization (EM):** Expectation maximization calculates the maximum likelihood of the parameter estimates when having missing data [19]. Therefore, expectation maximization is suitable for unlabeled data [21]. However, though EM algorithm is useful for finding local optimization when the size of missing data grows, the model would has slow convergence speed.

4. **Word2vec:** Comparing to other methods mentioned above, word2vec, which is introduced by Mikolov et al.in 2013, is relatively new, yet a popular approach for text classification problems. Word2vec helps to extract sentiment features and other semantic features from documents, and thus classifying the documents through these features [18]. Word2vec is suitable for classifying sparse data, such as text documents, and also increases the features that can be used.

Some popular automatic document classification features are as follows:

1. **Support vector machines:** Support vector machines (SVM) is a kind of supervised learning and usually used to solve pattern recognition problems, such as text categorization and face recognition [28]. Support vector machines is suitable for document classification because it is a linear

classifier, and document classification, in essence, is a linear problem; also, text vectors are sparse, which is the feature of vectors of SVMs [15].

2. **LSTM:** LSTM, also called Long Short-term Memory, is a kind of recurrent neural network, which is suitable for language modeling tasks. The used of LSTM in text classification tasks is growing mainly because of two reasons: first, the input length of a LSTM model is flexible which well-fitted to the arbitrary length of text files, such as social media posts, and news; second, LSTM has a forget gate which is useful for long text documents [27].

I will use word2vec to generate features for my stance classification model.

2.3 Stance classification

Stance classification, as a subarea of sentiment analysis, is to identify the stance of a particular target in a document [11]. With the popularity of the Internet, application scenarios of stance classification increase dramatically. For example, stance classification is usually applied in fact-checking tasks. Some researchers have combined stance classification and veracity predictions of news or social media posts [20].

My research will focus on stance classification of social media posts and responses, which can later be used in rumour veracity prediction in social

media.

2.4 Fact checking in social media

Fact checking is the task of determining the truthfulness. Generally, there are two types of fact checking: human fact-checking, and AI fact-checking. Human fact-checking can be of high accuracy, yet cost a good deal of labors and efforts; therefore, researches devoted to developing more advanced AI fact-checking algorithms.

According to Global social media research, the total number of social media users in the world is about 3.5 billion in 2019 [2]. With this number of social media users, false information spreads at an unprecedented speed in the social media. Therefore, fact checking in social media becomes an urgent task for researchers, companies, and governments.

Chapter 3

Rumour Evaluation Dataset

In this chapter, I will introduce the dataset used in this research, show some analysis results of the dataset, and give evaluation metrics for my model.

3.1 Dataset exploration

I used social media dataset of social media rumours for SemEval 2019 task 7 to build the model. All the posts are related to eight breaking news topics, which is collected through twitter. The dataset includes both the original tweets and the responses to those rumourous tweets. We also have true labels for these tweets, which are labeled as support, deny, query and comment.

In total, there are 297 original tweets together with 4,222 reply tweets, which is 4,519 tweets in total. My research mainly focuses on using the content of the tweets to predict its stance. The distribution of the tweet stances is listed in the Table 3.1.

From the table above, we can have an overview of the dataset:

1. The total record number of the training set is 4,519, while that of the testing set is 1,049. Therefore, train test split is about 4:1.

Table 3.1: Label distribution of training and test dataset

	Support		Deny		Query		Comment		Total
Train	910	20.1%	344	7.6%	358	7.9%	2,907	64.3%	4,519
Test	94	9.0%	71	6.8%	106	10.1%	778	74.1%	1,049
Total	1,004	18.0%	415	7.5%	464	8.3%	3,685	66.2%	5,568

2. The whole dataset is highly imbalanced. Comments account for 66.2% of the whole dataset, the percentages of which are 64.3 and 74.1 in training and testing sets respectively.

3. The label distributions in training and testing sets are different that, the proportion of the support tweets are about twice of that in the testing set.

According to the summary above, the size of data is relatively small, considering a machine learning task. Moreover, one thing we need to pay attention to is the imbalanced data issue.

3.2 Evaluation metrics

Evaluation metrics are measurements used to evaluate the performance of a machine learning model. In this study, I will use three metrics, confusion matrix, accuracy score, and Macro F1 to evaluate my models.

Table 3.2: Confusion matrix

	Actual class	
Predicted Class	True positive	False positve
	False negative	True negative

3.2.1 Confusion matrix

Confusion matrix is one of the most common metrics used to evaluate a classification model. A confusion matrix is as follows.

Table 3.2 gives an example of a confusion matrix.

1. **True positive:** The true label is positive and the predicted label is also positive.

2. **False positive:** The true label is negative and the predicted label is positive. False positive is also called type I error.

3. **False negative:** The true label is positive and the predicted label is negative. False negative is also called type II error.

4. **True negative:** The true label is negative and the predicted label is also negative.

3.2.2 Accuracy score

Accuracy score is defined as the percentage of the labels predicted correctly. After creating the confusion matrix, accuracy score can be calculated

through the following equation:

$$AccuracyScore = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$$

For my task, I will output five accuracy scores:

1. **Total accuracy:** The accuracy score of the whole dataset.
2. **Support accuracy:** The accuracy score of the posts and responses whose true labels are support
3. **Deny accuracy:** The accuracy score of the posts and responses whose true labels are deny
4. **Query accuracy:** The accuracy score of the posts and responses whose true labels are query
5. **Comment accuracy:** The accuracy score of the posts and responses whose true labels are comment

3.2.3 Macro F1

Macro F1 is a kind of F1 score recommended by the host of the challenge as the evaluation matrix. To understand what an F1 score is, we need to understand precision and recall first.

3.2.3.1 Precision

Precision is a measurement of the model accuracy, which evaluates the percentage of the true positive out of all actual labels. It can be defined as:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

3.2.3.2 Recall

Recall is a measurement of the model accuracy, which evaluates the percentage of the true positive out of all correctly predicted labels. It can be defined as:

$$Precision = \frac{TruePositive}{TruePositive + FalseNegative}$$

3.2.3.3 F1 score

From the former definitions of precision and recall, we can found that there is a trade-off between precision and recall, which is if we want to maximize the precision score, the recall will decrease; and vise versa. Therefore, F1 score is introduced to consider both precision and recall. F1 score is defined as:

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

From the formula above, we can see that F1 combines both precision and recall, therefore it's a good measurement to evaluate model accuracy in classification tasks.

There are two types of F1 score: macro F1 and micro F1. While macro F1 considers all classes equal, micro F1 gives each class weight according to the class distributions. In this study, I will use macro F1 as the main evaluation matrix, since it is encouraged by the challenge host.

Chapter 4

Baseline Exploration

The baseline model is provided by the sponsors of the Task 7: RumourEval 2019: Determining Rumour Veracity and Support for Rumours of the International Workshop on Semantic Evaluation 2019 [13], which is the winning solution of the "Task 8: RomourEval 2017 [9].

In this baseline model, Kochkina, Liakata, and Augenstein (2017) first extract text features using word2vec, and then combine with tweet lexicon to build a branch-LSTM model [16]. Their final accuracy score of the stance classification task is 0.782 in the development set, and 0.784 in the Testing set. And they published a git repository with instructions on reproducing this model `https://github.com/kochkinaelena/branchLSTM`.

The reason that I chose their solution as the baseline for my research is that this is the winning solution for the same task in 2017 . In order to reproduce baseline, I did three experiments, yet could not reproduce the same results as Kochkina, Liakata, and Augenstein (2017) reported. In the following sections, I will introduce the baseline model, describe my process of each experiment, analyze possible issues that cause the problems, and finally summarize my lessons learned.

4.1 Baseline model

As mentioned above, the baseline model is designed by Kochkina, Liakata and Augenstein (2017) [16]. Tweet stance classification has been a popular research topic, and many machine learning approaches has been customized [29] to solve this problem, such as Long-short term memory (LSTM) and some sequential data classification approaches.

Based on the characteristics of the tweet data, Kochkina, Liakata, and Augenstein (2017) [16] introduced a brand-new model, branch-LSTM, to solve the stance classification task. Figure 4.1 shows an example of the thread structure in the given dataset. For each source tweet, there could be several responses; and for each response, there also could be several responses. Therefore, it looks like a tree structure with branches. Kochkina, Liakata and Augenstein's model is designed based on the characteristics of this structure.

4.2 Experiments

After understanding the baseline model, I started to reproduce the baseline model. In this section, I will describe the implementation of each experiment, process, and analysis of the problems. The baseline model that I tried to reproduce can be found in the following git repository: `https://github.com/kochkinaelena/branchLSTM`.

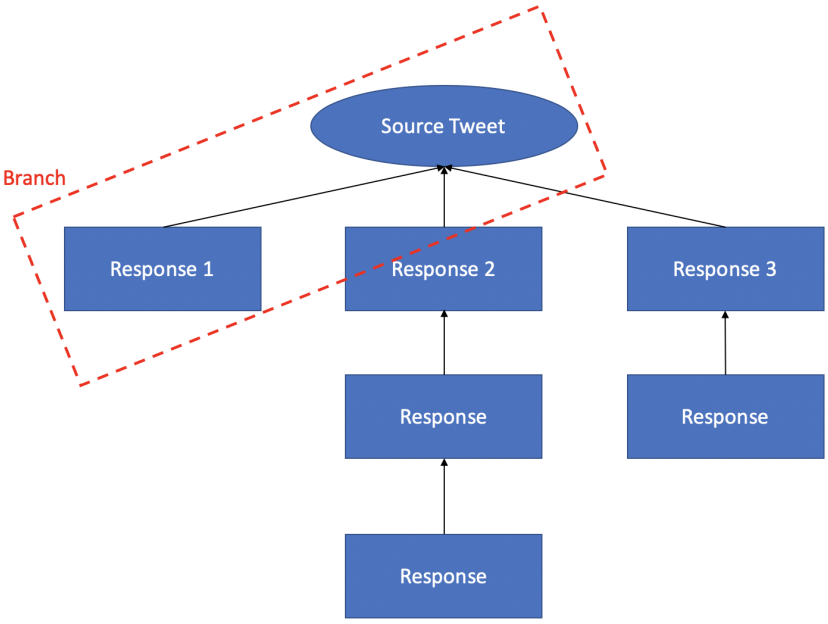

Figure 4.1: An example of the thread structure

4.2.1 Experiment I: running with a local machine

Kochkina gives detailed instruction to reproduce the result. In the instruction, they give two options: option one is installing on a local machine, and option two is installing on a Microsoft Azure VM with GPU. The author mentioned that the result could be reproduced using either a local machine or a virtual machine with GPU. Therefore, in my first experiment, I used a local machine without GPU, and the details of this machine can be found in Table 4.1.

After preprocessing the dataset, I got feature sets of word2vec and tweet lexicon. Then, according to the instruction, I ran the model with the

Table 4.1: Experiment I: environment

System version	Windows 10 Professional
Maker	Microsoft
Processor	Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz
RAM	8.00 GB

hyperparameters used in the paper and the features extracted. According to the instruction, if running with the optimized hyperparameters, the parameter search part will be skipped, and the process should take only a short time. However, I could not get any results after running the model and waiting for more than ten hours. I tried five times, but none of these worked.

I identified some reasons that may lead to the failure of my first experiment:

1. Different system environment: the authors use a virtual machine with GPU, while I use a local machine with much lower processing capability.

2. Version conflict of python: I installed both Python 2 and Python 3 in my local machine, and installed the required packages for both versions of Python. However, User-created versionPython 2 and Python 3 may not be compatible with the same version of packages, which could cause failures.

Table 4.2: Experiment II: environment

System version	Windows 10 Professional
Maker	Microsoft
Processor	Nvida TESLA K80 GPU
RAM	56.00 GB

4.2.2 Experiment II: hyperparameter searching

After experiencing the failure of my first experiment, I decided to try option two to install the solution on a Microsoft Azure VM with GPU, because the authors highly recommended us to use a virtual machine to reproduce the results. Moreover, with a GPU, I can run the hyperparameter search process to get a better understanding of the model. I used the Microsoft Azure data science virtual machine, and the details of this machine can be found in Table 4.2.

I chose the same NC6 virtual machine as Kochkina, Liakata, and Augenstein (2017) used for their branch-LSTM model. However, we used a different operating system: Kochkina, Liakata, and Augenstein (2017) used a Ubuntu server, while I was using a Windows system.

After configuring the environment, I started the implementation part. In order to keep consistent with my previous experiments, I used the saved features from Experiment I as my input data. Then I started the parameter search process. It took me roughly 5 hours, as it was mentioned in the instruction. However, the best parameter set I got is quite different as the optimized

Table 4.3: Hyperparameter search scope

	Parameter search scope
num_epochs	30, 50, 70, 100
num_dense_layers	1, 2, 3, 4
num_lstm_lyers	1, 2
l2reg	0, 0.0001, 0.0003, 0.001
rng_seed	364
learn_rate	0.0001, 0.0003, 0.001, 0.005
mb_size	32, 64, 100, 120
num_lstm_units	100, 200, 300
num_dense_units	100, 200, 300, 400, 500

hyperparameter set which Kochkina, Liakata, and Augenstein (2017) got. The hyperparameter search scope and comparison of two parameter sets are shown in Table 4.3 and Table 4.4 respectively.

From Table 4.3 and 4.4, we can see that there are nine hyperparameters in total. And the final optimized hyperparameter set I got is quite different with that in the baseline paper that only three parameters (num_epochs, rng_seed, and mb_size) are the same.

I considered the reason that may lead to the difference of the two sets of hyperparameters:

From Table 4.3, we can calculate the number of all the hyperparameter combination as follows:

$$HyperparameterCombination = \prod Hyperparameter$$

$$= 4 * 4 * 2 * 4 * 1 * 4 * 4 * 3 * 5 = 30720$$

Table 4.4: Comparison of hyperparameter sets

	Best stance parameters	My results
num_epochs	100	100
num_dense_layers	1	2
num_lstm_lyers	1	2
l2reg	0.0001	0.0
rng_seed	364	364
learn_rate	0.005	0.0003
mb_size	32	32
num_lstm_units	100	200
num_dense_units	400	500

So, it could be 30720 different hyperparameter combination at most. However, from the source code, we can see that we only conducted one hundred trials. Therefore, even the hyperparameter search scopes are the same, we could get different optimized hyperparameter set eventually. Moreover, considering from probability perspective, it would be rather tricky to get the same result.

4.2.3 Experiment III: results

After hyperparameter searching, I started to retrain the model with the hyperparameter set I obtained in the above steps. The results I obtained from two sets of hyperparameters in Table 4.4 are shown in Table 4.5.

As we can see from Table 4.5, using the best stance parameters, the accuracy score is much higher (about 14%) than using my results obtained from running parameter search myself. However, the F scores are quite close

Table 4.5: Results

	My results	Best stance parameters Kochkina, Liakata, and Augenstein (2017)
Accuracy	0.550	0.693
F score	0.231	0.238

Table 4.6: Model performance comparison

	Kochkina, Liakata and Augenstein (2017)	My result
Accuracy	0.782	0.693
F score	0.561	0.238

with only 0.07 difference.

As mentioned in the previous section, there could be 30720 possibilities of the hyperparameter combinations; and, using different hyperparameter combinations, the result could be varied a lot. However, even using the same parameter set as Kochkina, Liakata, and Augenstein (2017), I still could not obtain the same result. The model performance comparison of Kochkina, Liakata, and Augenstein (2017)'s result and my result is shown in Table 4.6.

As we can see from Table 4.6, both the accuracy and F score of my results are much lower than what Kochkina, Liakata and Augenstein (2017) has obtained, although we used the exact same parameter set.

I considered some reasons might lead to the failure of baseline reproduction:

1. Different system environment: Kochkina, Liakata, and Augenstein (2017) used "Ubuntu Server 16.04 LTS", while I used the Windows operating system. When the model first ran, there was an error message about CUDA (a required Python package) being not compatible with Windows system, and thus might resulting in different results.

2. Change of the implementation: since I used a different operating system as Kochkina, Liakata and Augenstein (2017), therefore I could not follow their instructions completely.

3. The baseline result is not easily reproducible: The author did mention running in the different system environment, the results may be slightly different. However, their records showed that the overall accuracy changed within 0.5% when using their best hyperparameter set, which does not apply to my experiment results. Therefore, I can reasonably posit that the baseline may not be easily reproducible.

4.3 Lessons learned

I spent over a month to reproduce the baseline, and through the process, my understanding of this task shifted a lot; therefore I wanted to include my reflection through the process. I will describe the importance of the baseline model, and summarize my lessons learned.

4.3.1 Why baseline models?

Baseline models are vital to machine learning tasks for several reasons:

1. Most machine learning tasks focus on improving the model accuracy, and the accuracy score is only significant when it is comparable with and better than other results. The primary role of the baseline model in the machine learning tasks is to give a comparable measurement for model performances.

2. Except providing a comparable measurement, baseline models also familiarize you with the dataset. It will eventually improve the efficiency in the following process because machine learning tasks depending highly on the feature engineering which requires a deep understanding of the whole dataset.

4.3.2 Lessons learned

4.3.2.1 Time management

Research can be extremely frustrating, especially when you are much behind schedule. At the beginning of this research, I planned to use two weeks at most to reproduce the baseline model. However, I were stuck in baseline reproduction for about one month, more precisely, four weeks. Accordingly, I had less time to focus on proceeding tasks for spending too much time in baseline reproduction. The whole thing teaches me that once I have a solid timeline of a project, I should stick to it or at least, making appropriate

changes.

4.3.2.2 Always have a plan B

Shifting research directions is acceptable, but you should always plan in advance. The failure of reproducing the baseline model finally resulted in my changing the former research direction. At first, I planned first to reproduce the baseline model, and then optimize the model by featuring engineering and parameter tuning. Apparently, without reproducing the baseline, I could not move to next steps and thus affecting my research progress. I should have considered the possible failure of my former research plan, prepared an alternative plan in advance.

Also, this is the nature of conducting research. It is often the case when you have some new findings, or you encounter some unsolvable difficulties during the research, do not be afraid to make some changes in your former plan.

4.3.2.3 Theory versus practice

This baseline exploration also made me think about the gap between professional knowledge and practical applications. Stanley (2001) defined "know how" as someone know how to do a certain thing in a certain way, which I formerly thought was professional knowledge [26]. As a master's student studying information science, I was confident that I knew how to reproduce the baseline with my professional knowledge obtained through my study.

However, this practice actually made me confused about what professional knowledge really is, and how can it help in practical situations.

Now I have a better understanding of "knowing how". Theory and practice supplemented each other. After systematic studies, one may acquire enough theoretical knowledge to involve in professions practice, which is professional knowledge, but this is not "knowing how". "Knowing how" should satisfied two conditions which are possessing the professional knowledge, and knowing how to use the professional knowledge in practical applications. Therefore, knowing how should be a combination of learning and practice.

Chapter 5

Concepts and models

In this chapter, I will describe some important concepts, and models used in this stance classification task.

5.1 Concepts

5.1.1 Supervised learning

Supervised learning, is a type of machine learning tasks. It uses labeled data to train the model and predicting classes or values, which is classification and regression respectively.

Generally speaking, there are two types of machine learning tasks: supervised learning, and unsupervised learning. Contrary to supervised learning, unsupervised learning deals with unlabelled data.

For this stance classification task, we have stances all the response tweets toward original tweets, and we train models using these stance labels; therefore, it is a supervised task.

5.1.2 Imbalanced data

As is mentioned in Chapter 3, the dataset is highly imbalanced. Imbalanced data may influence the performances of a model. Taking Logistic Regression as an example, its output is between 0 and 1. When the output of a sample is larger than 0.5, it will be predicted positive, and vice versa. Therefore, when the samples are imbalanced, using the default category standard may result in high accuracy yet high false positive or high false negative. Therefore, the model could be biased.

For example, supposed the depression rate of people is 5%, when predicting depression, we can simply predict all output as false, and still get a high accuracy rate of 95%. However, this result is not reliable.

There are several solutions to reduce the negative effect of the imbalanced data to the model performance.

1. **Parameter tuning:** Using parameter tuning to adjust threshold values, the model will be more sensitive to categories with fewer samples. For example, we can set different weights to different categories according to their sample distributions.

2. **Evaluation metrics:** Selecting a suitable evaluation metric is also an alternative solution. When dataset is highly imbalanced, instead of accuracy score, other evaluation metrics which are more sensitive to categories with fewer samples, such as F1 score and ROC (Receiver Operating Characteristic).

3. **Sampling:** Sampling is also frequently used to process imbalanced data. There are mainly two types of sampling: oversampling and undersampling. Basically, oversampling is adding random samples to categories with less samples, yet often leading to overfitting problems. Undersampling is randomly removing some samples from categories having larger samples. Although undersampling is convenient and easy to realize, it may accidentally delete some important information from the dataset.

Therefore, in this stance classification tasks, in order to reduce the effect of imbalanced data, I will use accuracy score, confusion and F1 score as evaluation metrics.

5.1.3 Overfitting and underfitting

Overfitting and underfitting are two common problems when training a machine learning model. I will mention these two concepts several times in the following parts of the report.

Overfitting generally refers to a model well fitted to the training data, yet performs less satisfactory in the testing data. In other word, the model is less capable of generalization. An underfitting model normally performs badly in both training and testing dataset.

To avoid overfitting and underfitting problems, selecting suitable models for a machine learning task is vital, which should consider all aspects of the dataset. For this stance classification tasks, considering the original dataset is

complicated enough, I will mainly focus on basic models.

5.2 Models

In this section, I will introduce some models used in this task. I selected these basic models mainly because they are frequently used in the classification tasks. Some researchers considered machine learning algorithms as "black box" because the logic behind the algorithms is difficult to explain, yet using basic models can help to build relatively explainable machine learning models.

5.2.1 SVMs

SVMs, also called support vector machines, first introduced by Vapnik and Chervonenkis in 1963 [25], is a supervised machine learning approach, which can be used in both classification tasks and regression analysis. Joachims (1988) believed SVMs are well fitted for document classification task because of the characteristics of the feature extracted and property representations [15]. SVMs are a really effective algorithm that it automatically extracted all the features without parameter tuning [15].

In 1963, SVMs are first developed to solve a linear problem, yet Boser, Guyon, and Vapnik proposed an approach of using SVMs to solve non-linear classification problems through kernels [5]. A simple understanding of SVMs is to find the maximum margin to classify data. The math behind the linear SVM classifier is explained as follows.

Suppose there is a line:

$$y(x) = w^T x + b$$

Then, the distance between any points and the line is:

$$\frac{1}{\|w\|}(w^t x_0 + b)$$

Or:

$$d = \frac{ax_1 + bx_2 + c}{\sqrt{a^2 + b^2}}$$

Therefore, the max margin can be calculated as follows:

$$\frac{1}{\|w\|} \min_n [y_i(w^T x_i + b)]$$

5.2.2 Logistic Regression

Logistic Regression, also called logit regression, is frequently used statistically. Using a sigmoid function, Logistic Regression is well suited to solve binary classification problems in machine learning. The formula of sigmoid function is as follows:

$$g(z) = \frac{1}{1 + e^{-z}}$$

The function curve of the sigmoid function is shown in Figure 5.1:

From Figure 5.1, we can see that the sigmoid function is an S shape curve, and its values are between 0 and 1, which is critical for a binary classification problem.

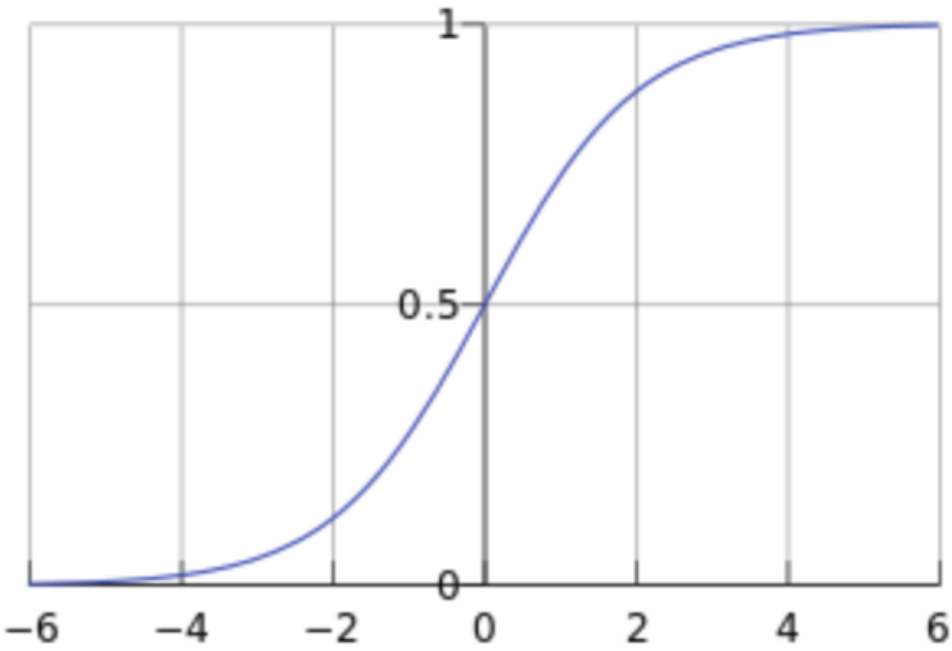

Figure 5.1: Sigmoid function

The cost function, also called loss function, is a measurement of model performances in machine learning. The cost function for Logistic Regression is as follows:

$$J(\Theta) = -\frac{1}{m}[\sum_{i=1}^{m}(y_i log h_\Theta x_i + (1-y_i) log(1-h_\Theta x_i))]$$

The function curve of the cost function is shown in Figure 5.2:

As we can see from Figure 5.2, when y = 1 and predicted value hypothesis function h=1, cost function J = 0; and vice versa. That's the math explanation of why Logistic Regression is good for 0-1 classification problems.

By introducing softmax function [7], we can use logistic regression to

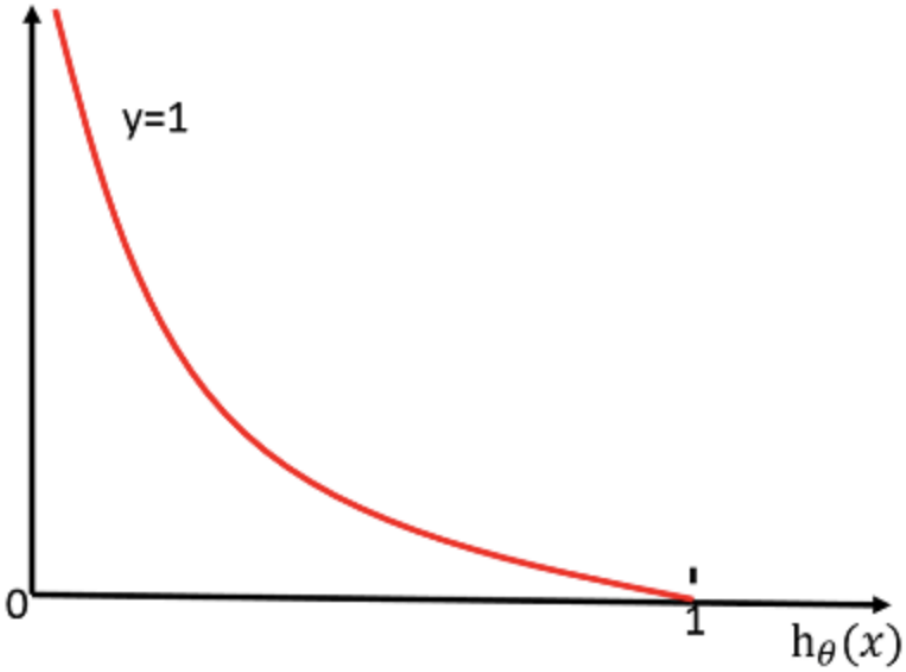

Figure 5.2: Cost function

solve multiclass classification problems. Softmax function can mapping the inputs into a value between 0 and 1.

5.2.3 Decision Tree

Decision tree, which is first introduced by Quinlan [22] in 1975, is a tree-structure model to split data with different features. An example of Decision Tree is shown in Figure 5.3. There are mainly three types of Decision Tree: ID3, C4.5, and CART.

1. **ID3:** ID3 algorithm calculates the entropy to decide parent nodes. The

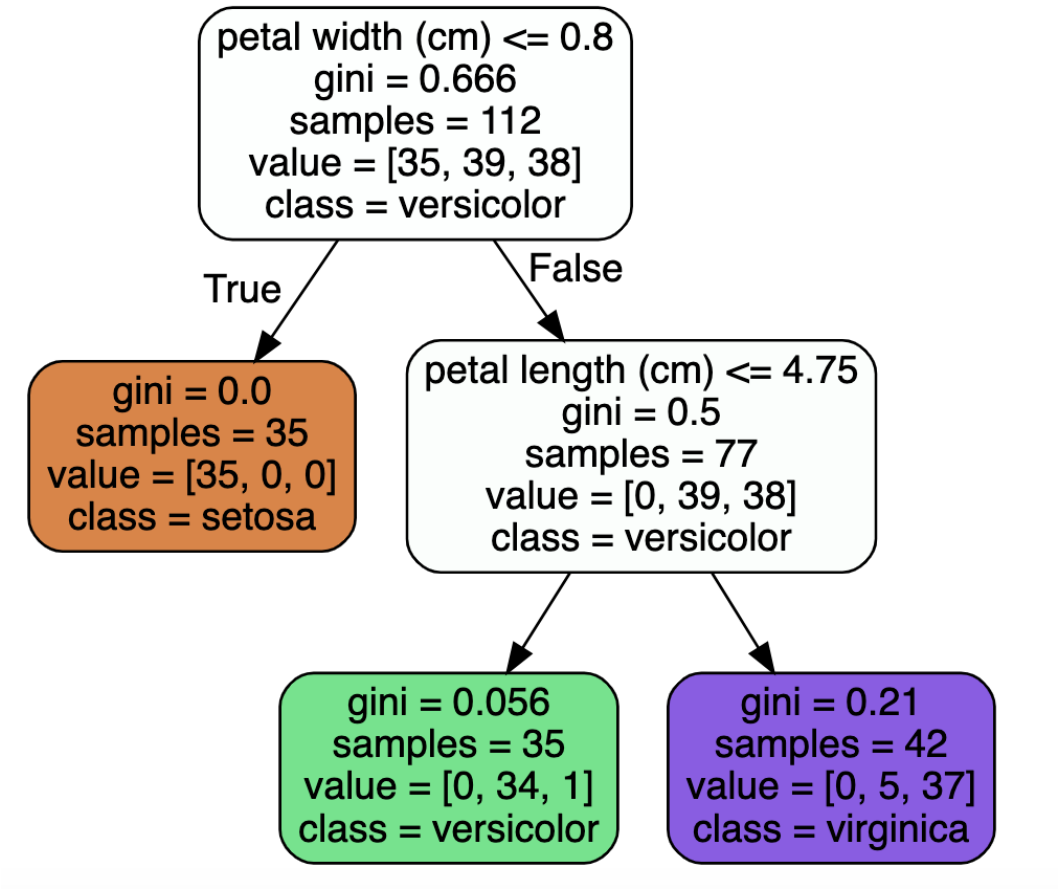

Figure 5.3: An example of the decision tree

larger the entropy is, better the classifier is. The formula of entropy is as follows [4]:

$$Entropy(x) = -\sum p(x) * log_2 p(x)$$

2. **C4.5:** A drawback for ID3 algorithm is overfitting since the algorithm will keep splitting data into smaller categories until reaching the stopping conditions. C4.5 overcomes this shortcoming by introducing information

gain. Information gain can be calculated as follows (D is dataset; (D—A) means data with feature A):

$$InformationGain(D, A) = Entrophy(D) - Entrophy(D|A)$$

Therefore, information gain means the reduced indeterminacy of information under condition A. The higher information gain is, the better the classifier is.

3. **CART:** CART is classification and regression tree [17], which calculates the parent nodes through GINI index. The lower the GINI index is , the better the classifier is. CART algorithm has the same disadvantage as ID3, which is overfitting. Pruning is used to solve this problem.

In my stance classification task, I will use C4.5 decision tree, since it solves the problem of overfitting.

5.2.4 Random Forest

Random Forest which is an ensemble learning model, was created by Tin Kam Ho [14]. In essence, Random Forest is made by decision trees. We have learned the theory of Decision Tree in the previous section. The primary advantage of the Random Forest over decision tree is that it does not have overfitting problems.

Random Forest does not merely average all the decision trees. When constructing trees, it takes random samples in the training dataset and creates notes with random feature sets.

Chapter 6

Experiments and results

In this chapter, I will introduce my research design, describe and analyze the results. Four classification models were used in the stance classification task: SVMs, Logistic Regression, Decision Tree, and Random Forest. I will present the results of these models and compare their performances using confusion matrix accuracy score, and Macro F1.

6.1 Experiments design

I designed two experiments to compare the performances of different classification models. All the models will use the same input features: word2vec and tweet lexicons. And the train test split is the same as the baseline model. The results are compared based on the models' performances in the test dataset.

In the first experiment, I ran four basic models: SVMs, logistic regression, decision tree and random forest, and comparing their performances by accuracy score and macro F1 score.

In the second experiment, I first selected a winning model from the first experiment, then comparing its results to the baseline model, and finally

discussed the advantages and disadvantages of both models.

6.2 Results

6.2.1 Experiment I

In this experiment, I ran Support Vector Machines, Logistic Regression, Decision Tree, and Random Forest, and comparing their performances by accuracy score and macro F1 score.

6.2.1.1 SVMs

In Table 6.1, we can see the results of SVMs. The accuracy is 52%, and macro F1 score is 0.220.

Table 6.1: Accuracy and macro F1 of SVMs

Accuracy	Macro F1
0.523	0.220

6.2.1.2 Logistic Regression

In Table 6.2, we can see the results of Logistic Regression. The accuracy is 53% and macro F1 score is 0.228.

Table 6.2: Accuracy and macro F1 of Logistic Regression

Accuracy	Macro F1
0.531	0.228

6.2.1.3 Decision Tree

In Table 6.3, we can see the results of the Decision Tree. The accuracy is 59% and macro F1 score is 0.223.

Table 6.3: Accuracy and macro F1 of Decision Tree

Accuracy	Macro F1
0.590	0.223

6.2.1.4 Random Forest

In Table 6.3, we can see the results of the decision tree model. The accuracy is 63%, and macro F1 score is 0.322.

Table 6.4: Accuracy and macro F1 of Random Forest

Accuracy	Macro F1
0.632	0.322

6.2.1.5 Summary

In Table 6.5, we can see the comparison of accuracy and macro F1 of SVMs, Logistic Regression, Decision Tree and Random Forest. All models use word2vec and tweet lexicon as input features.

Considering we were only using a basic model to classify the stances of the tweet, although the accuracy score is not quite high compared to 0.782 in the baseline model, it is acceptable.

Table 6.5: Results comparison

	Accuracy	Macro F1
SVMs	0.523	0.220
Logistic Regression	0.531	0.228
Decision Tree	0.590	0.223
Random Forest	0.632	0.322

The goal of this experiment is mainly to compare the performances of SVMs, Logistic Regression, Decision Tree, and Random Forest. The accuracy score of SVMs and logistic regression are quite close, which are 0.523 and 0.531 respectively. The accuracy of Decision Tree is much higher than SVMs and logistic regression (0.590). Random forest outperforms all other algorithms and achieves an accuracy score of 0.632. As mentioned in Section 1.3, Random Forest is the deserved winning solution, which outperformed SVMs, Logistic Regression and Decision Tree by 4%.

Random Forest is the only ensemble learning model [10] of the four. Ensemble is a machine learning method to combine different classifiers to improve the predicted performances. As we have mentioned in Chapter 5, Random Forest consist of a series of Decision Trees, and thus not surprisingly, it outperformed Decision Tree. Also, since ensemble learning selects the mode results of other models, it can help to reduce the effect of overfitting. Therefore, Random Forest became our winning solution.

6.2.2 Experiment II

In this section, I will compare the results of the Random Forest and the baseline model (branch-LSTM), and discuss their advantages and disadvantages

In Table 6.6, we can see the comparison of the results of Random Forest and Branch LSTM. Both models use word2vec and tweet lexicons as input features. The accuracy of Branch LSTM is much higher than our winning solution, Random Forest, by 6.1%.

Table 6.6: Results comparison of Random Forest and Branch LSTM

	Accuracy	Macro F1
Branch LSTM	0.693	0.238
Random Forest	0.632	0.322

Branch LSTM is an improved LSTM model proposed by Kochkina, Liakata, and Augenstein in 2017 to perform the stance classification tasks. In my experiment, although both models used word2vec and tweet lexicon as inputs, LSTM model receives more information since it takes all inputs as sequence data; so, these two models are incomparable. Therefore, I will discuss their advantages and disadvantages separately.

For Random Forest, its advantage of ensemble learning is non-negligible, which synthesizes the results of the decision trees. Also, the theory of Random Forest is easy-to-understand, which is well suited for students or researchers

first entering this area, who would like to have a more comprehensive understanding of the dataset and build an explainable machine learning model. However, you may have to remove some vital information if using Random Forest; since tweet threads are essentially sequence data, Random Forest is not able to take an inputs with time dimension. And lacking essential information may eventually lead to lower performances of the model.

For branch LSTM, it is a customized LSTM model which is well suited to tree-structure data, such as tweet threads. Also, It has the advantages of Long short-term memory (LSTM) model [12]; thus, using a branch LSTM, the information of social media posts can be fully exploited. Therefore, using branch LSTM, a much higher accuracy score than that of basic models can be achieved. However, because of the nature of neural networks, the algorithm is like a "black box" which is not explainable.

6.2.3 Conclusion

To summarize the results of two experiments, I mainly have two conclusions:

1. When using basic models to predict stance classes in social media, ensemble learning can be an excellent method because it not only combines advantages of other models and also helps to avoid overfitting.

2. Basic models and sophisticated models have its advantages and disadvantages, the characteristics of the dataset and the goal of the machine

learning tasks should be considered when selecting a suitable model for a specific task.

6.3 Future directions

Some future research directions are suggested as follows.

1. **Considering more features:** In order to build comparable models with baseline model, I kept the input features exactly the same (word2vec and tweet lexicon) as the baseline model. If time permitted, more features should be considered. There are many other alternative text features, such as N-grams [19] and TF-IDF [23], which could have improved model performances.

2. **Exploring more basic models:** Due to the limited time, my study only explores four basic models: SVMs, Logistic Regression, Decision Tree and Random Forest. There are other popular classification models, such as Naive Bayes [24] and Stochastic Gradient Descent [6], which may suit this stance classification task.

3. **Applying advanced algorithms:** To build an explainable model, I did not try any advanced algorithms except the baseline model. Some complex advanced models, such as C-LSTM [29], have been created to conduct a text classification task. I believe these advanced algorithms could outperform basic models in getting a higher accuracy score.

Appendix

Appendix 1

Reflections

In this part, I will reflect on the deficiencies of my research through the capstone process.

This study of stance classification in social media took about three months and a half. It mainly contained four stages: planning and research design, baseline exploration, experiment design, and result analysis, and report writing. The timeline for my capstone is shown in Table 1.1. I will present my reflections following the timeline of my capstone.

Table 1.1: Timeline of the capstone

Stages	Time Spent
Planning and research design	2 weeks
Baseline exploration	4 weeks
Experiment design and result analysis	4 weeks
Report writing	4 weeks

1.1 Planning and research design

I could have done better in time management. As we can see from Table 1.1, I spent four weeks in baseline exploration which accounted for almost 30%

of the whole process. If I could rearrange the time, I would spend at most two weeks in baseline exploration and focus more on experiment design and result analysis, so that I might implement more complex models and do more in-depth analysis.

1.2 Baseline exploration

My reflection of the process of the baseline exploration has been included in Chapter 4: Baseline exploration.

One thing I really wanted to emphasize is that the gap between professional knowledge and practice is more significant than I could ever imagine. Taking this stance classification task for example, I found a gap between my theoretical understanding of machine learning and real-life model implementation. However, beyond this, even though the model is implemented successfully, there still will be a gap between a machine learning model and its applications. Therefore, as a professional or someone who want to become an expert in this area, I should work on reducing the gaps like these.

1.3 Experiment design and result analysis

Some deficiencies of the experiment design and conclusions for this task are as follows:

1. The conclusion that ensemble learning models outperform basic models in stance classification task might be not sufficiently rigorous since I only

explored an ensemble learning, which may not be generalizable.

2. It could be presumptuous to compare the results of the baseline model and the random forest model since the inputs are not 100 percent the same. However, I have mentioned this drawback in the report, and thus I described the advantages and disadvantages of these models respectively.

3. As I have mentioned in the last part of the report, more features or more models can be tried for this stance classification task. Due to the limited time, my research is, in fact, superficial, which has lot of space to improve.

Bibliography

[1] Document classification, Sep 2018.

[2] Global social media research summary 2019, Apr 2019.

[3] Natural language processing, Mar 2019.

[4] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[5] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[6] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[7] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer, 1990.

[8] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175. Citeseer, 1994.

[9] Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1704.05972*, 2017.

[10] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[11] Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. Stance classification with target-specific neural attention networks. International Joint Conferences on Artificial Intelligence, 2017.

[12] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[13] Genevieve Gorrell, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. Rumoureval 2019: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1809.06683*, 2018.

[14] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[15] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[16] Elena Kochkina, Maria Liakata, and Isabelle Augenstein. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221*, 2017.

[17] Roger J Lewis. An introduction to classification and regression tree (cart) analysis. In *Annual meeting of the society for academic emergency medicine in San Francisco, California*, volume 14, 2000.

[18] Joseph Lilleberg, Yun Zhu, and Yanqing Zhang. Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 136–140. IEEE, 2015.

[19] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[20] An T Nguyen, Aditya Kharosekar, Saumyaa Krishnan, Siddhesh Krishnan, Elizabeth Tate, Byron C Wallace, and Matthew Lease. Believe it or not: Designing a human-ai partnership for mixed-initiative fact-checking. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, pages 189–199. ACM, 2018.

[21] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

[22] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[23] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[24] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[25] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[26] Jason Stanley and Timothy Willlamson. Knowing how. *The Journal of Philosophy*, 98(8):411–444, 2001.

[27] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[28] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[29] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.